Urban data is massive, heterogeneous, and spatio-temporal, posing a huge challenge for visualization and analysis. In this work, we design and implement visual analytics approach supports the visualization, querying, and exploration of urban data in an integrated visual interface. Our approach allows for cross-domain correlation from multiple data sources by leveraging spatial-temporal and social inter-connectedness features. Through our approach, the analyst is able to select, filter and aggregate across multiple data sources and extract information that would be hidden to a single data subset.

The VAUD query model has been designed to enable cross-domain queries and data fusion. In order to enable clear query specification, we split all queries into a series of atomic expressions. **query** is composed of three components: a query condition, a query operation, and queried results. The condition is a combination of four components (*which, when, where, what*) constraints Identification attributes (*which*), Spatial attributes (*where*), Temporal attributes (*when*), and Descriptive attributes (*what*)

An **extraction** is composed of three components, the queried results, an extraction operation, and a component of the object. An *extraction* indicates that a component is to be extracted from the object.

Object based: A list of objects can be extracted from each type of urban data, like the users from the mobile phone location data, the taxi drivers from the taxi GPS data. Each object consists of four distinct attributes: Identification attributes (*which*), Spatial attributes (*where*), Temporal attributes (*when*), and Descriptive attributes (*what*): We leverage the space-time-cube (STC) [18] as the canonical space for accommodating spatio-temporal objects.

Specifically, we split the entire time period of the urban data into slices (e.g., days). We construct an STC for each time slice and uniformly subdivide the STC into a 3D grid for a given resolution, where the resolution is determined based on the analysis tasks. As such, a cell of the STC refers to a geographical location and a time interval in the time slice associated with the STC. Finally, we sequentially relate each record of each object into an STC cell by leveraging the time stamp and location information. A reference to the object is then recorded in the cell. The urban data and associated STCs (Figure 2) support fast querying of spatio-temporal information and facilitate indirect connections of objects by means of the spatial-temporal inter-connectedness.

By assembling *atomic query* and *extraction*, comprehensive query operations can be executed to perform complicated tasks.

For example, if the analyst wants to find who rode a taxi that passed the central square the task, the analyst creates a query sequence that consists of three *atomic operations* and three *extraction operations*. First, the analyst needs to locate the central square, so the *which*!*ob jectPOI* expression is needed, where "*which*" contains "*id*=center square" condition, and the data source is the POI dataset. Once the position of the central square has been identified, the analyst can then specify a *where* ! *ob jectTaxi* expression to find taxis that pass the central square. After carefully studying all result candidates, the analyst determines which car best matches the specified query condition. Finally the analyst performs the expression *where+when* ! *ob jectPerson*, in which "*where+when*" denotes a spatio-temporal query of the taxi trajectory. This is needed to find the persons who rode in the taxi. The aforementioned process can be summarized as a query sequence presented in (Figure 4).

The VAUD interface consists of two main views: a scene view that shows the situational information as well as the properties of selected objects, and a query view that supports visual reasoning with intuitive sketch-based user interactions.
The query view is designed to support the construction of crossdomain query tasks by means of drag-and-drop interactions. We design two-tuple nodes and a directed Bezi´er curve representation to encode and organize components of queries and extracts.
The **condition node** allows the analyst to specify query conditions and data sources.
The **result node** presents the queried items and has a similar visual design as the condition node.
As the analysis becomes complex, both the condition node and result node can be folded to get a concise interface
In addition, the interface keeps a history action list to record the user's operations.
The scene view keeps a scene list to manage the ones shown on the scene view.

4/5
This case is aimed to locate the missing phone.

1. We want to explore MicroBlog posts and focuses on seeing what items have recently been noted as missing in the city. So we inputs "lost" as the keyword in a new condition node and then selects the Microblog dataset as the source and performs the query. The analyst explores the query results and selects an interesting post from 00:49 am. The post describes losing an iPhone: the blog writer took a taxi from SongTai Square to BaiHuaYuan early that morning and later realized she had misplaced her phone in the taxi.
2. the origin and destination (SongTai Square and BaiHuaYuan) were provided in the blog, To get the origin and destination geo-coordinates, the analyst performs a query where the locationdescription-tags of the origin and destination of the taxi ride from the post. the points-of-interest (SongTai Square and BaiHuaYuan) can be transformed into geocoordinates.
3. The Origin-Destination (OD) query from the POI dataset gives latitude and longitude, and the blog post provides a time of day. Next, to locate taxis that are near the points of interest in the early morning, an OD query is performed against the taxi's trajectory data to locate potential taxis the blog writer may have ridden in. By applying the intersection operator to the queried results, a set of taxis is detected. The analyst carefully studies the trajectories of each taxi in the scene view. and filters out taxis that were marked as occupied in the dataset.
4. Next, to further locate the taxi containing the missing phone, the analyst performs a spatio-temporal query over the mobile phone trajectory dataset to identify which phones were in the origin-destination vicinity. Finally, the analyst compares the trajectory of the taxis and the phones and identifies the taxi that is most likely to contain the missing phone.
5. The analyst studies the taxi profile data and finds the phone number of the driver who can then be contacted to ask if a cell phone was left in the car.